BT
02-27-02

Docket No.: G5030.0027/P027
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Motoyuki Kato, et al.

Application No.: 09/804,244-9167

Group Art Unit: 2171

Filed: March 13, 2001

Examiner: Not Yet Assigned

For: MEMORY ACCESS METHOD BY REFERENCING DATA, AND DATA PROCESSING DEVICE USING THE SAME

CLAIM FOR PRIORITY AND SUBMISSION OF DOCUMENTS

Commissioner for Patents
Washington, DC 20231

Dear Sir:

Applicant hereby claims priority under 35 U.S.C. 119 based on the following prior foreign application filed in the following foreign country on the date indicated:

Country	Application No.	Date
Japan	2000-069113	March 13, 2000

In support of this claim, a certified copy of the said original foreign application is filed herewith.

Dated: August 30, 2001

Respectfully submitted,

By _____

Thomas J. D'Amico
Registration No.: 28,371
DICKSTEIN SHAPIRO MORIN &
OSHINSKY LLP
2101 L Street NW
Washington, DC 20037-1526
(202) 828-2232
Attorneys for Applicant

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されてる事項と同一であることを証明する。
This is to certify that the annexed is a true copy of the following application as filed h this Office.

出願年月日
ate of Application: ２０００年　３月１３日

願番号
plication Number: 特願２０００－０６９１１３

願人
licant(s): オムロン株式会社

２００１年　４月　６日

特許庁長官
Commissioner,
Patent Office 及川耕造

【書類名】　特許願
【整理番号】　1305P
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/44
【発明者】
【住所又は居所】　京都府京都市右京区花園土堂町１０番地　オムロン株式会社内
【氏名】　加藤　意之
【発明者】
【住所又は居所】　京都府京都市右京区花園土堂町１０番地　オムロン株式会社内
【氏名】　柳　博之
【発明者】
【住所又は居所】　京都府京都市右京区花園土堂町１０番地　オムロン株式会社内
【氏名】　中川　伸二
【発明者】
【住所又は居所】　京都府京都市右京区花園土堂町１０番地　オムロン株式会社内
【氏名】　馬場　洋介
【特許出願人】
【識別番号】　000002945
【氏名又は名称】　オムロン株式会社
【代理人】
【識別番号】　100078916
【弁理士】
【氏名又は名称】　鈴木　由充
【手数料の表示】
【予納台帳番号】　056373

【納付金額】　　21,000円

【提出物件の目録】

【物件名】　　明細書　1

【物件名】　　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9803438

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　インタプリタ型言語によるプログラムの実行方法およびその方法を用いた情報処理装置

【特許請求の範囲】

【請求項１】　インタプリタ型言語によるプログラムの実行に先立ち、前記プログラムからメモリに対するアクセス先を特定するための参照用の情報を抽出してその情報による参照を解決するとともに、得られた参照解決結果を前記参照用の情報を介してプログラムにリンクさせて保存しておき、

所定の情報を参照してメモリにアクセスすることを要求するプログラムの実行に際し、その参照用の情報を介して前記プログラムにリンクする参照解決結果に基づき前記メモリに対するアクセス先を特定することを特徴とするインタプリタ型言語によるプログラムの実行方法。

【請求項２】　インタプリタ型の言語によるプログラムが組み込まれた情報処理装置において、

前記プログラムを実行するプログラム実行手段と、

メモリに対するアクセス先を特定するための参照用の情報を介して、それぞれその情報による参照解決結果をプログラムにリンクさせて保存する参照解決結果登録手段とを具備し、

前記プログラム実行手段は、所定の情報を参照してメモリにアクセスすることを要求するプログラムの実行に際し、その参照用の情報を介して前記プログラムにリンクする参照解決結果に基づき前記メモリ内のアクセス先を特定する情報処理装置。

【請求項３】　前記プログラムは、バイトコードの実行用プログラムと、この実行用プログラムにリンクして前記参照用の情報の内容を示す情報とから成り、前記参照解決結果登録手段は、前記参照解決結果を前記実行用プログラムへのリンク情報内に保存する請求項２に記載された情報処理装置。

【請求項４】　前記実行用プログラムへのリンク情報は、複数の固定長のコード情報を含み、前記参照解決結果は、筆頭のコード情報の定められた位置に格納される請求項３に記載された情報処理装置。

【請求項５】　前記実行用プログラムおよびそのリンク情報は、プログラム実行時にＲＯＭより読み出される請求項３または４に記載された情報処理装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】

この発明は、インタプリタ型言語によるプログラムの実行方法およびこの種プログラムの組み込まれた情報処理装置に関し、特にこの発明は、情報の参照によりメモリへのアクセス先を特定する必要のあるプログラムを実行するための技術に関連する。

【０００２】

【従来の技術】

サン・マイクロシステムズ社が開発したオブジェクト指向プログラム言語「Java」（同社保有の登録商標）は、バイトコードに変換されることにより、プラットフォームに依存せずに、Javaインタプリタを有する実行環境で動作することから、種々の機器に組み込む情報処理システムを設定するのに適した言語として注目を集めている。

【０００３】

このJavaのプログラムによる情報処理システムでは、メモリへの不正なアクセスを禁止するために、プログラム上にメモリのアクセス先を直接示さずに、名前の情報によりアクセス先を間接的に示すようにしている。

たとえば変数（フィールド）に値を設定したり、その設定値を読み出すことを指示するプログラムでは、その変数を、変数の属するクラス名，変数名，および変数の型名（int型，byte型など）により指定する。またメソッドの呼出しを行うプログラムでは、呼出し対象のメソッドを、そのメソッドの属するクラス名やメソッド名により指定する。一般に、所定の規則（名前を示す文字情報）を用いて処理対象（変数，メソッド，クラスなど）についての情報格納位置を探す処理を「参照」と呼び、その参照を指示することを「参照要求」という。（たとえば、変数，メソッド，クラスのそれぞれについての参照を、「フィールド参照」「メソッド参照」「クラス参照」という。）

【０００４】

「仮想マシン」と称されるJavaインタプリタは、所定の名前の情報による参照を要求するプログラムの実行に際し、その名前の情報により所定の参照用テーブルを検索して（詳細は後記する）、参照すべき情報が格納されたアドレスを特定するようにしている。

このように指定された情報による参照を行って、情報格納位置を特定することを「参照解決」と呼んでいる。

【０００５】

図６は、Javaによるプログラムのコンパイル結果の例を示す。なお図中、塗りつぶしされた各矩形は、一単位分のプログラムコードであって、各コード内にはそれぞれ複数の情報が含まれている。図示例では、個々の情報を白抜きの矩形内に示してある。

図６中、１０は、変数に値を代入することを指示するメッセージ（アセンブラコードで"putstatic Sample.value:int"と表される。）を変換して得たバイトコードの実行用プログラムであって、指示内容を表すインストラクションコード（Ｂ３）およびインストラクションが指定する参照の内容の記述先のアドレスを示すオペランド（００　０６）とから構成される。（以下このインストラクションコード付きの実行用プログラムを「インストラクション」という。）

【０００６】

このインストラクション１０には、図中、右側に示すような構成の参照用情報がリンクする。この情報は、先頭に情報の種類を表すタグが付された可変長のコード情報を複数組み合わせた構成をとる。各コード情報には、それぞれ「コンスタントプールエントリ番号」と称される識別番号（以下「エントリ番号」と略す）が付与されるとともに、そのエントリ番号を介したリンクが設定されている。前記オペランドには、この参照用情報の筆頭のエントリ番号６が書き込まれており、各コード情報間のリンクをこの筆頭のエントリから順に辿ることにより、ソースプログラムに記述されたクラス名，変数名，変数の型名の各情報を示す文字列を得ることができる。

【０００７】

なおこの参照用情報は、クラス単位で保存される。（以下、クラス単位での参照用情報のことを「クラスファイル情報」という。）図６では、説明を簡単にするために、１インストラクションについての参照用情報を抜き出しており、図示されていないエントリ番号は、他のインストラクションの参照用情報に使用されている。

【０００８】

図中、エントリ番号２０にはクラス名"Sample"を表す文字情報が、エントリ番号２７には変数名"Value"を表す文字情報が、それぞれ登録される。またエントリ番号１７には、変数の型名のint型を表す文字列"I"が登録される。なおこれらのエントリ番号２０，２７，１７の文字列情報はエンコードされており、そのエンコード情報の前には文字列のバイト数を表すコード情報および文字列のエンコード情報の登録であることを示すタグ「０１」が付けられる。

【０００９】

エントリ番号１，６，９の情報は、前記文字列の情報が何を参照するための情報であるかを示している。

筆頭のエントリ番号６の情報は、「フィールド参照」という意味のタグ「０９」により、以下の情報がフィールド参照用の情報であることを示す。このエントリ番号６には、エントリ番号１および９を示すコード情報が登録される。

【００１０】

エントリ番号１には、「クラス参照」と言う意味を持つタグ「０７」が付与され、前記クラス名を表す文字情報の登録先であるエントリ番号２０へのリンクが設定される。またエントリ番号９には、「名前・型参照」という意味を持つタグ「０Ｃ」が付与され、前記変数名の文字情報が登録されたエントリ番号２７および変数の型名が登録されたエントリ番号１７へのリンクが設定される。

【００１１】

このようなデータ構成により、インストラクション１０のオペランドに書き込まれたエントリ番号６から順に参照情報内のリンクを辿ることにより、フィールド参照のために必要なクラス名，変数名，型名の各情報を得ることができる。他のコマンドについても同様に、ソースプログラムのコンパイルにより、指定され

た参照の具体的な内容を示す参照用情報にリンクするインストラクションが生成される。

【００１２】

なお同じ参照を要求する複数のインストラクション間では、参照用情報を共有することができる。たとえばプログラムの複数位置に前記インストラクション１０が存在したり、このインストラクション１０と同様のフィールド参照を要求する他のインストラクション（たとえば"getstatic Sample.Value:int"というプログラムより得られるインストラクション）が存在する場合、これらのインストラクションのオペランドには、それぞれ前記エントリ番号６が書き込まれて、前記図６の参照用情報へのリンクが設定される。

【００１３】

インタプリタは、各インストラクションを実行する際に、前記インストラクションコードからコマンドの内容を認識するとともに、オペランドに書き込まれたエントリ番号から順にリンクを辿って、指定された参照の内容やその参照に用いる名前の情報を認識する。

またシステム内には、各クラスで設定された変数やメソッドについて、それぞれの名前の情報からメモリ内における変数やプログラムの格納位置を特定するための参照用のテーブルが設定されている。インタプリタは、前記参照用情報から得た名前の情報により前記参照用のテーブルを参照して、前記インストラクションの指定するメモリへのアクセス先を得るための参照解決処理を実行する。

【００１４】

図７は、フィールドの参照解決に用いられるクラステーブルおよびフィールドテーブル（図中、１１，１２で示す）の構成を示す。なお図７において、１３は、プログラム上に設定される各オブジェクトの作業領域として機能するヒープメモリ、１４は、クラス名や変数名などの文字列を格納するための文字列テーブルである。

【００１５】

前記クラステーブル１１およびフィールドテーブル１２には、前記クラスファイル情報の作成に伴い、そのクラスについての情報が登録される。クラステーブ

ル１１には、クラストップアドレス，ストリングトップアドレス，自クラスのネームインデックス，親クラスのネームインデックス（スーパーインデックス），クラス内のオブジェクトのサイズ，アクセスフラグなどの情報が登録される。（以下これらの登録情報を「クラス情報」と総称する。）

【００１６】

クラストップアドレスは、前記クラスファイル情報における筆頭データの格納先のアドレスを示す。ストリングトップアドレスとは、文字列テーブル１４において、各クラスに含まれるクラス名や変数名などの文字列情報を格納する領域の先頭アドレスを示す。ネームインデックスとは、前記参照用情報において各クラス名を表す情報のコンスタントプールエントリ番号である。またアクセスフラグとは、クラスへのアクセスの可または不可を表すためのものである。

これらのクラス情報には、それぞれクラス毎に固有のインデックス（以下これを「クラステーブルインデックス」という）が付与される。

【００１７】

フィールドテーブルには、プログラム上で設定された変数毎に、オブジェクトオフセット，クラステーブルインデックス，ネームインデックス，タイプインデックス，アクセスフラグなどの情報が登録される。（以下、これらの登録情報を「フィールド情報」と総称する。）

【００１８】

前記オブジェクトオフセットは、前記変数を含むオブジェクトに付与された作業領域において、前記変数に割り当てられるフィールドの相対位置を示す値である。この値は、クラスファイル情報の作成時に、各クラスのインスタンスがヒープメモリ内に確保すべき作業領域の大きさが決定され、その作業領域内で各変数の格納領域が割り当てられたときに決定される。

ただしstatic変数については、その変数の値そのものがフィールドテーブル内に保存される。

【００１９】

クラステーブルインデックスとは、前記クラステーブルにおいて、前記変数を有するクラスのクラス情報に付与されたインデックスである。ネームインデック

ス，タイプインデックスは、それぞれ前記参照用情報において変数名や型名を表す情報のコンスタントプールエントリ番号であり、アクセスフラグは、前記と同様に、その変数へのアクセスの可、不可を示す情報である。

各フィールド情報には、個別にインデックス（以下これを「フィールドテーブルインデックス」という）が付与される。

【００２０】

インタプリタは、フィールドの参照解決を行うにあたり、まず前記インストラクションにリンクする参照用情報から得たクラス名によりクラステーブルを検索して、そのクラスのクラステーブルインデックスを取得する。つぎにフィールドテーブルを検索して、前記クラステーブルインデックスを含む情報の中から変数名および型名が前記参照用情報より得た情報と一致するフィールド情報を抽出する。そしてそのフィールド情報内のオブジェクトオフセットの値から、この変数に割り当てられた格納領域のアドレスを認識する。

【００２１】

上記のフィールドの参照解決結果を得ると、インタプリタは、再度同じインストラクションを実行する場合に備えて、前記インストラクション内のインストラクションコードを解決済みのコードに書き換えるとともに、オペランドを、前記フィールド参照により取得したフィールドテーブルインデックスの番号に書き換える。この書換え処理により、つぎにこのインストラクションを実行する際は、オペランドに書き込まれたフィールドテーブルインデックスから直ちにフィールドテーブル内のオブジェクトオフセットを読み出して、指定された変数の格納位置を得ることができる。

【００２２】

なおメソッドの参照解決のためには、クラス名やメソッド名からメソッドのプログラムの格納位置などを導くための参照用テーブル（メソッドテーブル）が設定されている。インタプリタは、インストラクションおよびこれにリンクする参照用情報から読み取ったメソッド名によりメソッドテーブル内のインデックスを取得することでメソッドの参照解決を行うとともに、再度のインストラクションの実行に備えて、そのインストラクションのオペランドを前記取得したメソッド

テーブルインデックスに書き換えるようにしている。

【００２３】

図８は、前記Javaのソースプログラムから生成されたバイトコードのプログラムによるシステムにおいて、システムの起動からプログラムの実行までの一連の処理の手順を示す。

このシステムを構成する実行用プログラムやクラスファイル情報は、装置内のＲＯＭに記憶されており、電源の供給によりシステムが立ち上がると、実行用プログラムがＲＡＭにロードされた後、起動用のプログラムによって最初に起動させるクラスが指定される（ＳＴ１）。これを受けて、つぎのＳＴ２では、指定されたクラスのクラスファイル情報がＲＡＭ内に読み出され、さらにＳＴ３では、前記クラスファイル情報が読み出されたクラスについての情報が、クラステーブル，フィールドテーブル，メソッドテーブルなどの参照用テーブル内に登録される。

【００２４】

このようにしてインストラクションを実行する環境が整うと、ＳＴ４では、最初のインストラクションが読み込まれる。ここでこのインストラクションがまだクラスファイル情報を作成していないクラスの参照を要求するものであれば、ＳＴ５からＳＴ６に移行してそのクラスのクラスファイル情報の作成を指定する。そしてその指定に応じてＳＴ２，３でクラスファイル情報の作成や参照用テーブルへのエントリが完了してから、再度、前記インストラクションの読込みが行われる。

【００２５】

インストラクションにより参照要求されたクラスのクラスファイル情報が既に作成されている場合は、そのクラスファイル情報からクラス名，変数名，メソッド名などの名前の情報を認識する。このインストラクションがシステムの立上げ後に初めて実行される場合、そのインストラクションコードは未解決のコードであるから、ＳＴ７からＳＴ８に進むことになる。

【００２６】

ＳＴ８では、前記インストラクションのオペランドに書き込まれたエントリ番

号からクラスファイル情報のリンクを辿ってクラス名や変数名（あるいはメソッド名）を認識した後、認識した名前により各種参照用テーブルを参照して、変数の格納位置やメソッドのプログラムの格納位置などを特定する参照解決を行う。そしてＳＴ９で、前記参照解決により得たアドレスにアクセスして、インストラクションを実行した後、ＳＴ１０からＳＴ４に戻り、以下、インストラクション毎に同様の処理が実行される。

なおＳＴ８での参照処理の解決に伴い、実行されたインストラクションのインストラクションコードやオペランドが書き換えられるから、つぎにこの書換え済みのインストラクションが読み込まれたときは、ＳＴ７からＳＴ９に進んで、参照解決を行うことなく、前記インストラクションを実行することができる。

【００２７】

【発明が解決しようとする課題】

上記の手順によれば、システム起動後に各インストラクションを初めて実行する際には、参照解決を必要とするすべてのインストラクションについてＳＴ８の処理を実行する必要がある。この参照解決処理では、前記したように、変数やメソッドの格納先を特定するために何段階もの参照を行わなければならないため、インストラクションが読み込まれてから実行されるまでにかなりの時間がかかる。このためシステムが起動した直後のプログラム実行時に、処理速度が極端に遅くなる。またシステム起動直後のプログラム実行時間と参照が解決された２回目以降のプログラム実行時間とが異なることから、プログラム実行時間の予測が必要なリアルタイム処理用の機器での利用が困難となっている。

【００２８】

しかも従来のシステムでは、一連のプログラムの複数位置に同様の参照を要求するインストラクションが存在しても、インストラクション毎に同じ参照解決を繰り返さなければならないので、参照解決の効率が著しく悪い、という問題もある。

【００２９】

さらにインストラクションコードやオペランドを書き換える都合上、実行用プログラムをＲＡＭ内に記憶する必要があるが、この場合、装置立ち上げ時にプロ

グラムをＲＡＭ内にロードするための時間が必要となり、システムの立上げに時間がかかる。また実行用プログラムの格納のために、ＲＯＭと同容量のＲＡＭを用意しなければならないので、組込み機器のコスト高を招く、という問題も発生する。

【００３０】

この発明は、上記問題点に着目してなされたもので、プログラムを実行する前に、メモリに対するアクセス先を特定するための参照用の情報を含むプログラムについての参照解決を行っておき、各参照解決結果をプログラムにリンクさせて保存しておくことにより、プログラム実行時に参照用の情報から直ちにアクセス先を特定して、プログラムの実行時間の安定化ならびに高速化を実現し、リアルタイム性が要求される機器に適した情報処理を実現することを第１の目的とする。

【００３１】

またこの発明は、プログラムのＲＡＭへのロードを不要とすることにより、システムの起動に要する時間を短縮するとともに、ＲＡＭの容量を大幅に削減して機器の製作コストを削減することを、第２の目的とする。

【００３２】

さらにこの発明は、参照解決結果をプログラムとともにＲＯＭから読み出してプログラムを実行することにより、ＲＡＭの容量をさらに削減するとともに、この種システムの組込み機器において、システムの起動後すぐにプログラムの高速実行を開始できるようにすることを、第３の目的とする。

【００３３】

【課題を解決するための手段】

請求項１の発明では、インタプリタ型言語によるプログラムの実行に先立ち、前記プログラムからメモリに対するアクセス先を特定するための参照用の情報を抽出してその情報による参照を解決するとともに、得られた参照解決結果を前記参照用の情報を介してプログラムにリンクさせて保存しておく。そして所定の情報を参照してメモリにアクセスすることを要求するプログラムの実行に際し、その参照用の情報を介して前記プログラムにリンクする参照解決結果に基づき前記

メモリに対するアクセス先を特定するようにしている。

【００３４】

請求項２の発明は、インタプリタ型の言語によるプログラムが組み込まれた情報処理装置において、上記の方法を実施するために、前記プログラムを実行するプログラム実行手段、メモリに対するアクセス先を特定するための参照用の情報を介して、それぞれその情報による参照解決結果をプログラムにリンクさせて保存する参照解決結果登録手段とを具備させる。プログラム実行手段は、所定の情報を参照してメモリにアクセスすることを要求するプログラムの実行に際し、その参照用の情報を介して前記プログラムにリンクする参照解決結果に基づき前記メモリ内のアクセス先を特定する。

【００３５】

上記インタプリタ型言語によるプログラムとは、たとえばJavaのソースプログラムからコンパイルされたバイトコードのプログラムの場合、変数名やメソッド名などの参照を要求する実行用プログラム（インストラクション）と、このインストラクションにリンクして参照用の情報の具体的な内容（クラス名，変数名，メソッド名など）を示す情報とにより構成される。プログラム実行手段は、インタプリタの機能により実現するもので、前記のリンク情報により参照用の情報の内容を確認しつつ各プログラムを順に実行する。

【００３６】

前記参照用の情報による参照解決とは、参照用の情報により所定の情報を参照してメモリへのアクセス先を特定し、以後の参照処理を不要となす処理をいう。

たとえば「フィールドの参照解決」では、プログラムに示されるクラス名や変数名などの文字情報に基づき、指定された変数の格納先を表すアドレス、またはそのアドレスに一意に対応づけられた情報（前記したフィールドテーブルインデックスなど）を取得する。また「メソッドの参照解決」では、メソッド名などの文字情報に基づき、指定されたメソッドのプログラムの格納先を表すアドレス、またはこのアドレスに一意に対応づけられた情報（前記したメソッドテーブルインデックスなど）を取得する。また「クラスの参照解決」においては、クラス名を表す文字情報に基づき、指定されたクラス内のメソッドや変数に関する情報（

前記したクラス情報）の格納先を表すアドレス、またはこのアドレスに一意に対応づけられた情報（前記したメソッドテーブルインデックスなど）を取得する。

【００３７】

請求項３の発明では、前記プログラムがバイトコードの実行用プログラムと、このプログラムにリンクして参照用の情報の内容を示す情報とから構成される場合に、前記参照解決結果を前記実行用プログラムへのリンク情報内に保存する。さらに請求項４の発明では、リンク情報は複数の固定長のコード情報を含んでおり、前記参照解決結果は、筆頭のコード情報の定められた位置に保存される。

【００３８】

請求項５の発明では、バイトコードの実行用プログラムおよびそのリンク情報をＲＯＭから読み出して、プログラムを実行するようにしている。

【００３９】

【作用】

請求項１の発明によれば、プログラムの実行前に、そのプログラム上でメモリに対するアクセス先を特定するための参照用の情報について、それぞれその情報により得られる参照解決結果をプログラムにリンクさせて保存するので、プログラムの起動直後でも、保存された参照解決結果を用いて速やかにメモリにアクセスし、指定された処理を実行することが可能となる。またプログラムの実行時間が一定になるので、実行時間の予測が可能である。

【００４０】

また参照解決結果を、参照用の情報を介してプログラムにリンクさせるから、参照解決に伴ってのプログラムの書換えが不要となり、システム起動時にプログラムをＲＡＭへロードする必要がない。

またこのようにプログラムを書き換えずに、同じ参照用の情報の参照を要求するプログラムについて、同じ参照解決結果をリンクさせるから、従来のように同様の参照解決を繰り返す必要がない。

【００４１】

請求項２の発明によれば、各参照用の情報についてあらかじめ得た参照解決結果が、システムの立ち上げと同時にプログラムにリンクさせた状態で提供され、

プログラムの起動後すぐに高速の処理を実行することが可能となる。

【００４２】

請求項３の発明では、参照解決結果は、実行用プログラムの指定する参照用の情報の内容を示すリンク情報内に保存されるので、本来のプログラムのリンク構造を利用して参照解決結果をプログラムにリンクさせることができる。

【００４３】

請求項４の発明によれば、参照解決結果は、リンク情報の筆頭のコード情報の特定位置に格納されるので、参照解決結果を簡単に取り出すことが可能となる。またプログラム実行前には、各情報ファイルの参照解決結果の格納位置をチェックすることにより、参照未解決の情報を簡単に判別することが可能となる。

【００４４】

請求項５の発明によれば、参照解決結果を含むリンク情報は、実行用プログラムとともにＲＯＭに格納されるので、システムの立ち上げ直後にプログラムを高速で実行することが可能となる。

【００４５】

【実施例】

図１は、この発明の一実施例にかかる情報処理システムの構成を示す。

この情報処理システムは、コンピュータが組み込まれた機器内において、Javaのソースプログラムよりコンパイルされたバイトコードのプログラム（以下単に「プログラム」という）を実行するもので、インストラクション格納部１，クラスファイル情報格納部２，インストラクション実行部３，テーブル管理部４，参照解決部５，ヒープメモリ６，メモリ割り当て部７などにより構成される。なおこれら構成のうち、インストラクション実行部３，テーブル管理部４，参照解決部５，メモリ割り当て部７の各処理部は、Javaインタプリタにより実現する。またインストラクション格納部１はコンピュータのＲＯＭ内に設定され、クラスファイル情報格納部２，ヒープメモリ６，およびテーブル管理部４により作成される各種参照テーブルは、それぞれＲＡＭ内に設定される。

【００４６】

前記インストラクション格納部１には、バイトコードのプログラムのうち、前

記したインストラクション、すなわちインストラクションコードとオペランドとから成る実行用のプログラムが格納される。

【００４７】

クラスファイル情報格納部２には、各インストラクションについて指定された参照の内容を具体的に示す参照用情報をクラス毎にまとめたクラスファイル情報が設定される。クラス内の各参照用情報は、それぞれ前記したエントリ番号に基づいてリンクする複数の情報により表される。またインストラクション格納部１内で参照要求を行うインストラクションのオペランドには、それぞれそのインストラクションの指定する参照用情報の筆頭のエントリ番号が書き込まれている。

【００４８】

テーブル管理部４は、各クラスのクラスファイル情報より従来と同様のクラステーブル，フィールドテーブル，メソッドテーブルなどの参照用テーブルを作成する。これら参照用テーブルの設定情報は、必要に応じてインストラクション実行部３，参照解決部５，メモリ割り当て部７に供給される。

【００４９】

インストラクション実行部３は、インストラクション格納部１より各インストラクションを順に読み出して実行する。

メモリ割り当て部７は、インストラクション実行部３の指示に応じて、プログラムの実行により生成されたインスタンスについての作業エリアをヒープメモリ６内に設定するとともに、前記オブジェクトオフセットの値に基づき、各インスタンスの具備する変数に、該当する作業エリア内の所定アドレスを割り当てる。

【００５０】

参照解決部５は、インストラクションの要求する参照を解決するためのものである。この実施例では、プログラムの実行に先立ち、各クラスファイル情報に含まれる名前の情報について、それぞれその名前による参照を解決し、得られた参照解決結果をクラスファイル情報内に書き込むようにしている。

なお参照解決の手法は、前記した従来の方法と同様であって、例えばフィールド参照の解決結果としてフィールドテーブルインデックスが、メソッド参照の解決結果としてメソッドテーブルインデックスがそれぞれ特定され、クラスファイ

ル情報内に書き込まれる。またクラス参照については、指定されたクラス名に該当するクラスのクラステーブルインデックスが取得され、参照解決結果としてクラスファイル情報内に保存される。

【００５１】

図２は、この実施例におけるクラスファイル情報の構成であって、前記図６に示した従来の構成と同様に、１インストラクションの参照用情報を抽出して示す。

この実施例でも、従来と同様に、「フィールド参照」のエントリ番号６のコード情報を筆頭にして複数のコード情報がリンクして、クラス名，変数名，変数の型名の各文字情報、およびその文字情報が何の参照のための情報であるかが示されている。

【００５２】

これらのコード情報は、固定長に設定される。各コード情報の先頭には、従来と同様のタグが格納され、その他の情報は末尾詰め（図示例によれば右詰め）に格納されて、中間部分に空白部（図中「－」で示す）が設定される。

【００５３】

なおこの実施例では、各コード情報を固定長にする必要上、名前を表す文字列の情報は、別途設けられる文字列テーブルに格納している。前記文字列テーブルは、クラス単位で設定されており、１７，２０，２７の各エントリ番号には、各文字列の位置データとして、文字列テーブルのテーブル番号（図中の０）と、そのテーブルにおける該当文字列の位置（図中、１０００，１００４，１０１０の各数字で示す）とが格納される。

【００５４】

前記参照解決結果は、前記参照用情報の筆頭エントリの定められた空白部に格納される。図示例では、前記参照用情報のリンクの筆頭にあたるエントリ番号６の空白部には、フィールドの参照解決により得られたフィールドテーブルインデックス（以下「解決されたフィールドテーブルインデックス」という）が格納される。またこのエントリ番号６からクラス名の格納先にリンクするエントリ番号１の空き領域には、前記フィールドの参照解決の過程で得られたクラステーブル

インデックス（以下「解決されたクラステーブルインデックス」という）が格納される。なお１，６以外のエントリ番号の空き領域は、空のまま保持される。

上記の空き領域においてインデックスを格納する位置は固定されており、この変数名によるフィールド参照が解決しているか否かを、簡単に判別することができる。

【００５５】

なおクラスファイル情報内のメソッド名を表す参照用情報については、メソッドの参照解決により得られたメソッドテーブルインデックス（以下「解決されたメソッドテーブルインデックス」という）が、同様に、前記参照用情報の筆頭エントリに書き込まれる。

【００５６】

図３は、図１の構成の情報処理システムにおいて、システム起動からプログラム実行に至るまでの一連の手順を示す。

図中、ＳＴ１～３では、従来と同様にして、指定されたクラスのクラスファイル情報が作成された後、クラステーブル，フィールドテーブル，メソッドテーブルなどの参照用テーブルが設定される。

つぎのＳＴ４～６では、前記参照解決部５により、各クラスファイル情報における変数名やメソッド名による参照が解決される。そしてすべてのクラスファイル情報に対する参照解決が完了するまで、ＳＴ６で参照未解決のクラスを順に指定し、参照解決を実行する。すべてのクラスのクラスファイル情報に対する参照解決が完了すると、ＳＴ５が「ＮＯ」となり、以下、ＳＴ７，８のループで各インストラクションが順に実行される。

【００５７】

図４は、前記図３のＳＴ４の詳細な手順を示す。なおここでは説明を簡単にするためにフィールドの参照解決によりフィールドテーブルインデックスを取得する手順を示している。

【００５８】

以下、図４の流れに沿って、前記図２のクラスファイル情報についてのフィールドの参照解決を行う場合の手順について説明する。

参照解決部５は、まずＳＴ４－１で、クラスファイル情報からフィールド参照のタグ「０９」の付いた情報であって、解決されたフィールドインデックスが含まれていない参照未解決の情報（エントリ番号６）を抽出する。さらにＳＴ４－２では、ＳＴ４－１で前記抽出した情報からクラス名参照の情報（エントリ番号１）およびそのリンク情報（エントリ番号２０）を順に辿って、前記変数の所属するクラス名を取得する。

【００５９】

ここで参照解決部５は、取得したクラス名を有するクラスがクラステーブルに登録されているか否かをチェックし、登録されている場合は、そのクラスのクラステーブルインデックスを取得する（ＳＴ４－３，４－５）。なお前記クラスがクラステーブルに登録されていない場合は、ＳＴ４－４で、そのクラスを前記クラスファイル情報に基づきクラステーブルおよびフィールドテーブルに登録した後に、クラステーブルインデックスを取得する。

【００６０】

さらに参照解決部５は、前記ＳＴ４－１で抽出したエントリ番号６の情報から名前・型参照の情報（エントリ番号９）およびそのリンク情報（エントリ番号２７，１７）を順に辿って変数名および型名を取得する（ＳＴ４－６）。そしてフィールドテーブルを検索して、前記ＳＴ４－５で取得したクラステーブルインデックスを具備し、かつ変数名と型名とがＳＴ４－６で取得したものと一致するフィールド情報を抽出し、この情報に付されたフィールドテーブルインデックスを取得する（ＳＴ４－８）。

【００６１】

前記ＳＴ４－５およびＳＴ４－８でフィールドテーブルインデックス，クラステーブルインデックスの各インデックスを得ると、参照解決部５は、前記したクラスファイル情報内の特定の空き領域にこれらのインデックスを書き込む（ＳＴ４－９）。

【００６２】

以下同様にして、フィールド参照のタグ「０９」の付いた参照未解決の情報を順に抽出し、この情報にリンクする情報を辿って得たクラス名，変数名および型

名により、フィールドテーブルインデックス，クラステーブルインデックスを取得し、各インデックスの値をクラスファイル情報に書き込んでいく。

なおフィールドテーブルの検索において、クラステーブルインデックス，変数名，型名の組合せが一致するフィールド情報が見つからなかった場合は、ＳＴ４－７が「ＮＯ」となってＳＴ４－１１に移行し、リンクエラーとして処理される。

【００６３】

このようにしてプログラムの実行前に、プログラム上の各変数について、それぞれその変数によるフィールドの参照解決を行って、解決されたフィールドテーブルインデックスを、そのフィールドの参照を指定するインストラクションの示すエントリ番号の空白部に格納する。これによりインストラクション実行部３は、前記インストラクション１０の実行時に、そのインストラクション１０のオペランドに書き込まれたエントリ番号から前記解決されたフィールドテーブルインデックスを取得し、そのインデックスの値から得られるオブジェクトオフセット値により認識したフィールドにアクセスして、前記インストラクションを実行することになる。

【００６４】

なおクラス参照を要求するインストラクションには、そのクラスのクラス名の文字情報を含む参照用情報（前記したタグ「０７」付きのエントリ番号１およびこのエントリ番号にリンクしてクラス名の文字列を示すエントリ番号２０より成る）がリンクし、その筆頭のエントリ番号の空白部に、前記解決されたクラステーブルインデックスが書き込まれている。したがってインストラクション実行部３は、オペランドに書き込まれたエントリ番号を介して前記解決されたクラステーブルインデックスを取得し、そのインデックスに対応するクラス情報を読み出すことにより、クラス内のオブジェクトやメソッドに関する情報を簡単に得ることができる。

【００６５】

同様にメソッド参照を要求するインストラクションにも、プログラム実行に先立つ参照解決により、そのインストラクションにリンクする参照用情報の筆頭の

エントリ番号に、解決されたメソッドテーブルインデックスが書き込まれる。したがってこのインストラクションを実行する際には、そのインストラクションのオペランドのエントリ番号から取得したメソッドテーブルインデックスにより、メソッドのプログラム格納先を認識し、即座にメソッド呼出しを実行することができる。

【００６６】

またクラス内に文字列を表すStringオブジェクトを生成するインストラクションが含まれる場合、そのオブジェクト名についても、最初のインストラクション実行時に参照解決を行ってその解決結果をクラスファイル情報内に書き込んでおくことにより、Stringオブジェクトの参照を要求するインストラクションを高速で実行することができる。

【００６７】

このStringオブジェクトについては、そのオブジェクトの表す文字列を格納する文字列テーブルと、オブジェクトの所属するクラスや前記文字列テーブル内における文字列の格納位置などを示すオブジェクトテーブルとが設定され、インスタンスの生成に応じて、前記文字列テーブルとオブジェクトテーブルとの双方に、相互にリンクする情報が登録される。したがって参照解決の際には、前記と同様に、インストラクションにリンクする参照用情報の筆頭のエントリ番号に、参照解決により得られたオブジェクトテーブルインデックスを書き込むことにより、以後は、Stringオブジェクトの参照要求に応じて、クラスファイル情報から指定されたオブジェクトのオブジェクトテーブルインデックスを読み出した後に、速やかに文字列テーブル内の文字列の格納位置を取得し、該当する文字列情報にアクセスすることができる。

【００６８】

このように上記の情報処理システムでは、名前の情報の参照により所定のアドレスへのアクセスを指定するインストラクションを実行する前に、クラスファイル情報から各参照用情報を抽出してその参照処理を解決するとともに、参照解決の結果をインストラクションのオペランドにリンクする参照用情報の読出し開始位置に保存しておくので、プログラムが起動した直後から、各インストラクショ

ンについて、クラスファイル情報内の参照解決結果を用いてアクセス先のアドレスを一意に特定し、インストラクションを高速で実行することが可能となる。また第１回目のプログラム実行時間と、２回目以降のプログラム実行時間との間に差違が生じないから、プログラム実行時間を予測することが可能となる。

【００６９】

なおインストラクションを書き換えずに、そのオペランドに記述された筆頭エントリ内に参照解決結果を書き込むので、従来のリンク構造を利用して参照解決結果をプログラムにリンクさせることができる。これにより参照解決結果を用いてインストラクションを実行するには、インストラクション実行部３を、オペランドに記述された筆頭エントリから参照解決結果を読み出すように設定するだけで良く、インタプリタの大幅な設計変更を行う必要がない。

しかも参照解決結果は、筆頭エントリの特定位置に書き込まれているので、参照解決結果を簡単に読み出すことができる。また参照解決時に参照未解決の情報を探す処理も、簡単化できる。

【００７０】

またこの実施例の情報処理システムでは、インストラクションの書換えが不要であるため、インストラクションをＲＯＭから読み出して実行することにより立ち上げ時のプログラムのロードが不要となって、機器への電源投入後にシステムを速やかに立ち上げることができる。またプログラムのロードが不要となることから、ＲＡＭの容量を大幅に削減することができる。

しかも前記の参照解決結果を、クラスファイル情報内の各参照用情報に対応づけて保存することにより、同じ情報の参照を要求するインストラクション間で参照解決結果を共有できるので、同様の参照解決を繰り返し行う必要がなく、参照解決処理の効率を向上することができる。

【００７１】

さらに上記のクラスファイル情報内に各参照用の情報についての参照解決結果を保存する方法を応用して、あらかじめ参照解決結果が書き込まれた状態のクラスファイル情報や、このクラスファイル情報に応じた参照用テーブルを作成し、これらのクラスファイル情報や参照用テーブルをＲＯＭ内に格納することも可能

である。この方法によれば、システムの立ち上げ後の参照解決は不要となり、すぐにプログラムを起動させて高速の処理を実行することができるので、特に小型でリアルタイム性を要求する機器に対し、有用な情報処理システムを提供することができる。

【００７２】

図５は、この発明の第２実施例にかかる情報処理システムの構成を示す。

この実施例の情報処理システムは、前記図１の構成のうち参照解決部５を除く各構成を具備するが、クラスファイル情報格納部２およびテーブル管理部４による各種参照用テーブルは、インストラクション格納部１とともにＲＯＭ内に設定される。なおその他の構成については、前記第１の実施例と同様である。

【００７３】

クラスファイル情報格納部２には、インストラクション格納部１に格納されたすべてのクラスについてのクラスファイル情報が保存され、テーブル管理部４は、これらのクラスの情報が登録されたクラステーブル，フィールドテーブル，メソッドテーブル，文字列テーブルなどを管理する。

またクラスファイル情報格納部２には、各クラスの変数名，メソッド名，クラス名について、あらかじめ実行された参照解決により得られた解決結果を示すインデックス情報が、前記第１の実施例と同様の形式で格納されている。

【００７４】

上記構成の情報処理システムでは、電源投入後、直ちにシステムが起動するとともに、参照解決結果の書き込まれたクラスファイル情報や各参照用テーブルも設定され、各インストラクションを参照解決の結果に基づいて高速で実行することができる。しかもこの情報処理システムは、容量の大きなＲＡＭを必要としないから、安価なコストで高速の情報処理が可能な組込み機器を提供することが可能となる。

【００７５】

【発明の効果】

請求項１，２の発明では、プログラムの実行に先立ち、メモリに対するアクセス先を特定するための参照用の情報による参照処理を解決し、その参照解決結果

を参照用の情報を介してプログラムにリンクさせて保存するので、プログラムを起動した後は、保存された参照解決結果を用いて速やかにメモリにアクセスするが可能となり、プログラムの実行速度の安定化かつ高速化を実現することができる。またプログラムの実行時間が一定となるため、プログラムの実行時間の予測を必要とするリアルタイム処理用の機器でも利用可能な情報処理システムを提供することができる。

【００７６】

また参照解決に応じてプログラムを書き換える必要がないので、プログラムのＲＡＭへのロードが不要となり、電源投入からシステムの起動までに要する時間を短縮できる上、ＲＡＭの容量を大幅に削減することができる。しかも同じ参照用の情報を参照するように指示されたプログラムについての参照解決を一度で済ませることができるので、短時間で参照解決を終了してプログラムの実行に移行することができる。

【００７７】

請求項３の発明では、本来のプログラムのリンク構造を利用して参照解決結果をプログラムにリンクさせることが可能となるので、インタプリタの大幅な設計変更が不要となる。

【００７８】

請求項４の発明によれば、プログラムの実行時にリンク情報の筆頭のコード情報の特定位置から参照解決結果を効率よく取り出すことができるので、プログラムの実行速度をさらに高速化できる。またプログラム実行前の参照解決時にも、前記特定位置に参照解決結果が書き込まれているか否かをチェックすることにより、参照未解決の情報を簡単に判別して、参照解決処理の高速化を実現することができる。

【００７９】

請求項５の発明によれば、プログラムの実行時に、実行用プログラムおよびこれにリンクする参照解決結果をＲＯＭより読み出すので、電源投入後、すぐにプログラムを起動させて高速の処理を実行することができる。しかもプログラムや参照解決結果をＲＡＭにロードする必要がないから、ＲＡＭの容量をさらに削減

することができ、安価なコストで高速の情報処理が可能な組込み機器を提供できる。

【図面の簡単な説明】

【図１】

この発明の一実施例にかかる情報処理システムの構成を示す機能ブロック図である。

【図２】

インストラクションとクラスファイル情報との構成および両者の関係を示す図である。

【図３】

図１の情報処理システムにおける処理手順を示すフローチャートである。

【図４】

参照解決の詳細な手順を示すフローチャートである。

【図５】

情報処理システムの第２の構成を示す機能ブロック図である。

【図６】

従来のインストラクションとクラスファイル情報との構成および両者の関係を示す図である。

【図７】

参照用テーブルの構成を示す図である。

【図８】

従来の情報処理システムにおける処理手順を示すフローチャートである。

【符号の説明】

１　インストラクション格納部

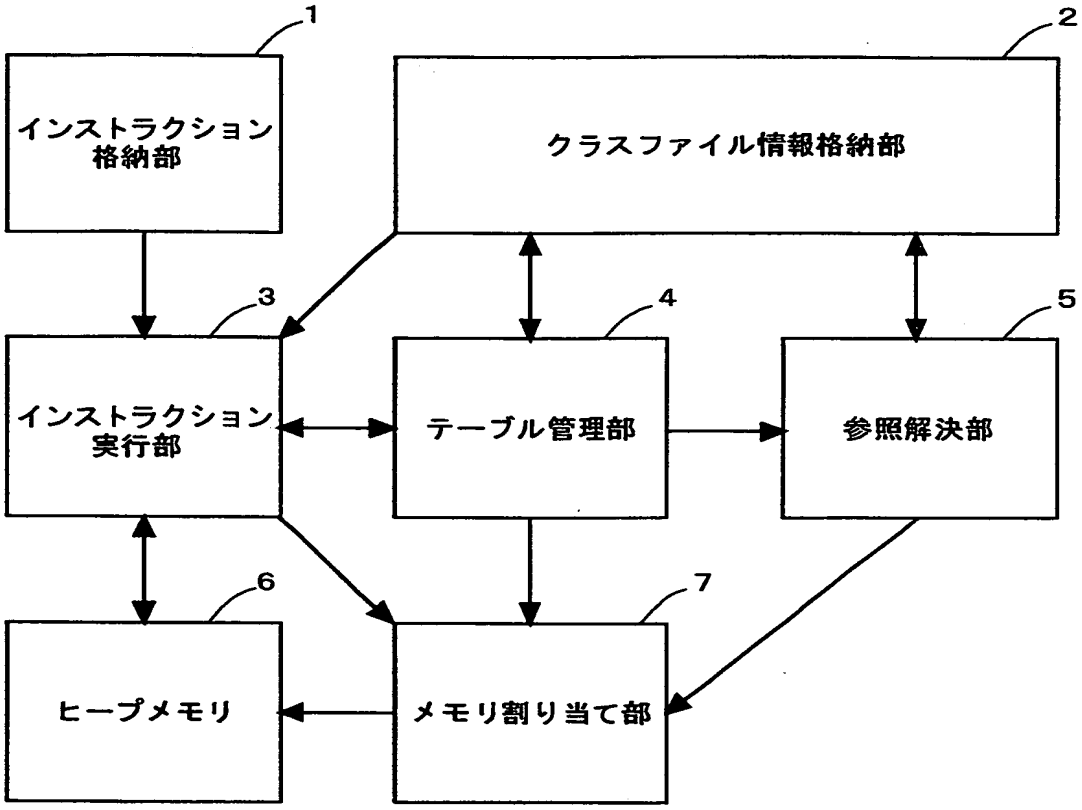
２　クラスファイル情報格納部

３　インストラクション実行部

５　参照解決部

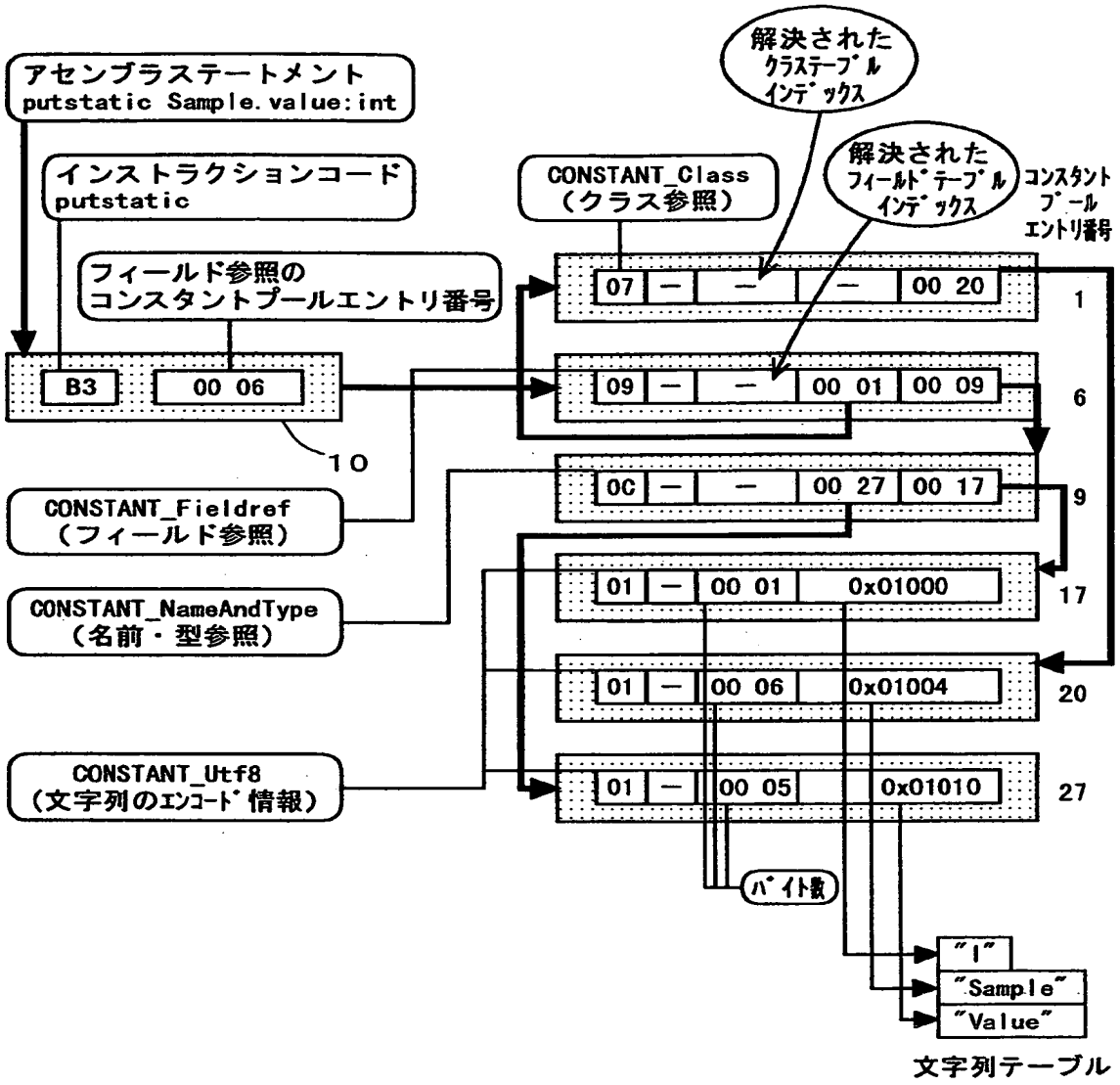
【書類名】　　図面

【図１】

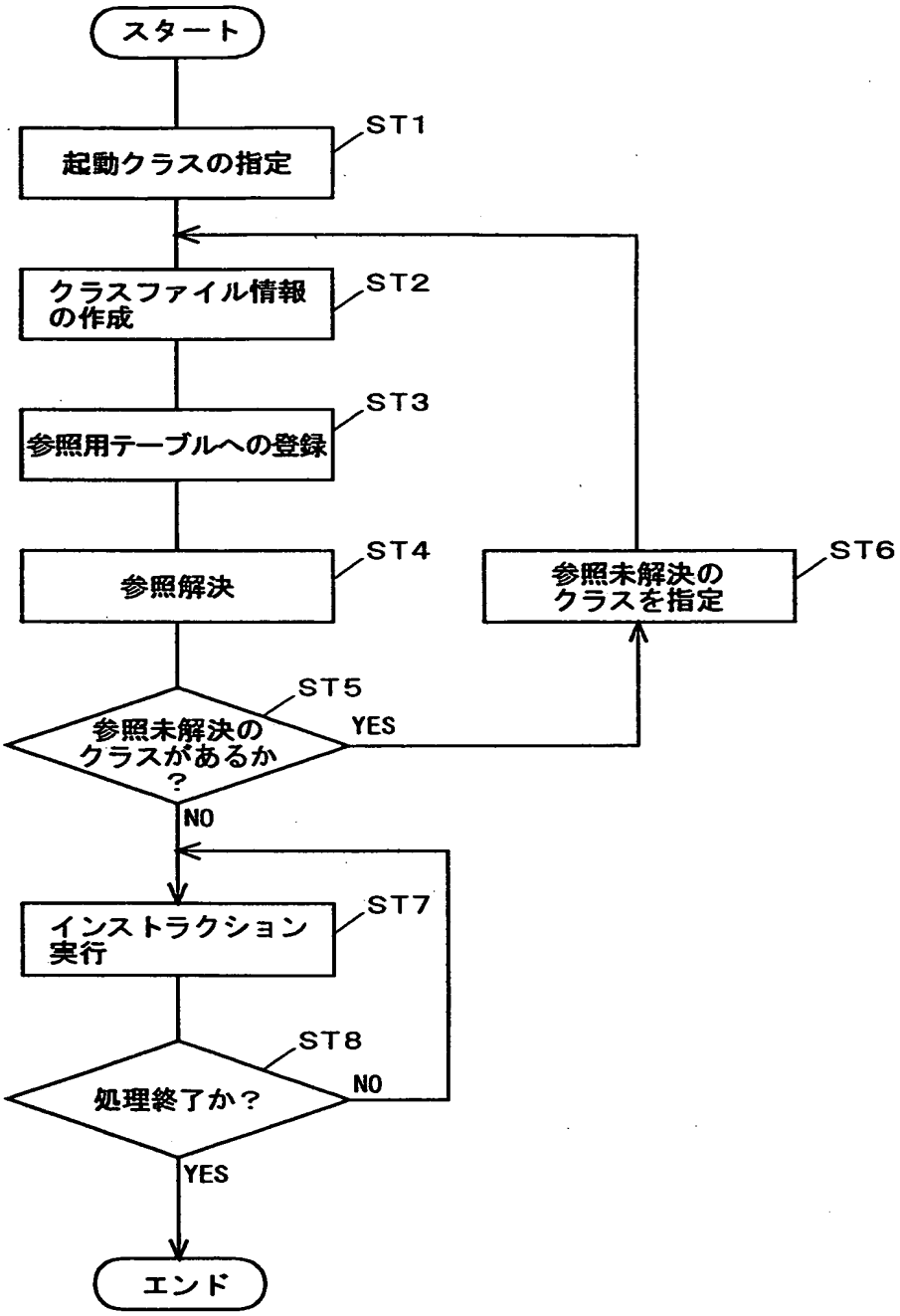


出証特２００１－３０２５８７６

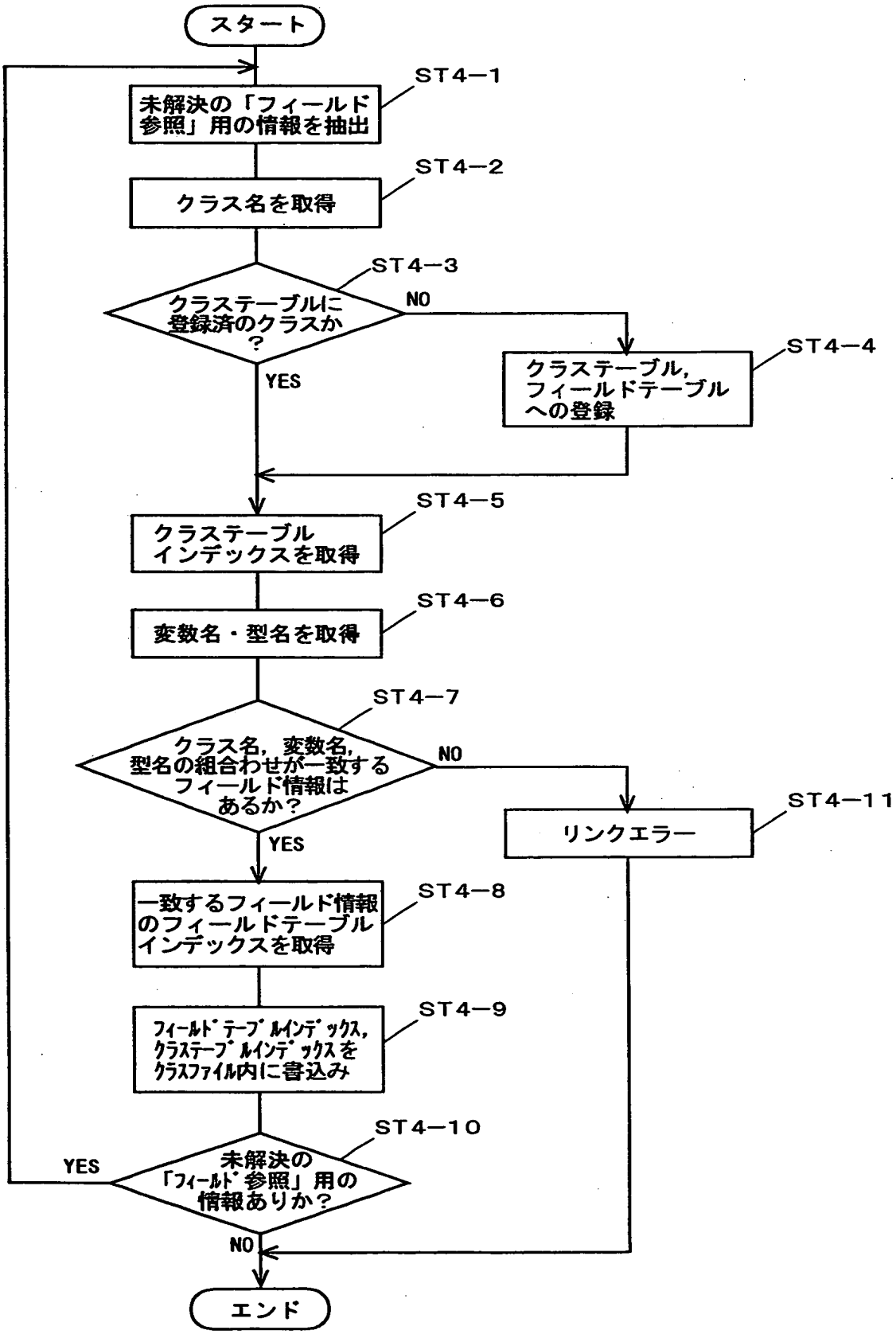
【図２】


アセンブラステートメント
putstatic Sample.value:int
インストラクションコード
putstatic
フィールド参照の
コンスタントプールエントリ番号
B3
00 06
10
CONSTANT_Fieldref
（フィールド参照）
CONSTANT_NameAndType
（名前・型参照）
CONSTANT_Utf8
（文字列のｴﾝｺｰﾄﾞ情報）
解決された
ｸﾗｽﾃｰﾌﾞﾙ
ｲﾝﾃﾞｯｸｽ
解決された
ﾌｨｰﾙﾄﾞﾃｰﾌﾞﾙ
ｲﾝﾃﾞｯｸｽ
CONSTANT_Class
（クラス参照）
コンスタント
ﾌﾟｰﾙ
エントリ番号
07 — — — 00 20
1
09 — — 00 01 00 09
6
0C — — 00 27 00 17
9
01 — 00 01 0x01000
17
01 — 00 06 0x01004
20
01 — 00 05 0x01010
27
ﾊﾞｲﾄ数
"I"
"Sample"
"Value"
文字列テーブル

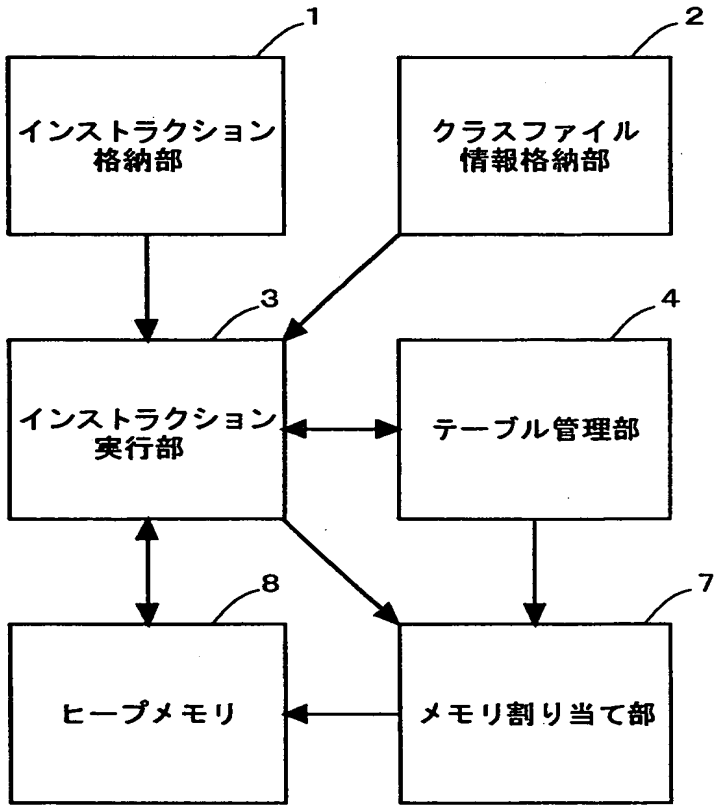
【図３】


スタート
起動クラスの指定
ST1
クラスファイル情報の作成
ST2
参照用テーブルへの登録
ST3
参照解決
ST4
参照未解決のクラスを指定
ST6
参照未解決のクラスがあるか？
ST5
YES
NO
インストラクション実行
ST7
処理終了か？
ST8
NO
YES
エンド

【図４】

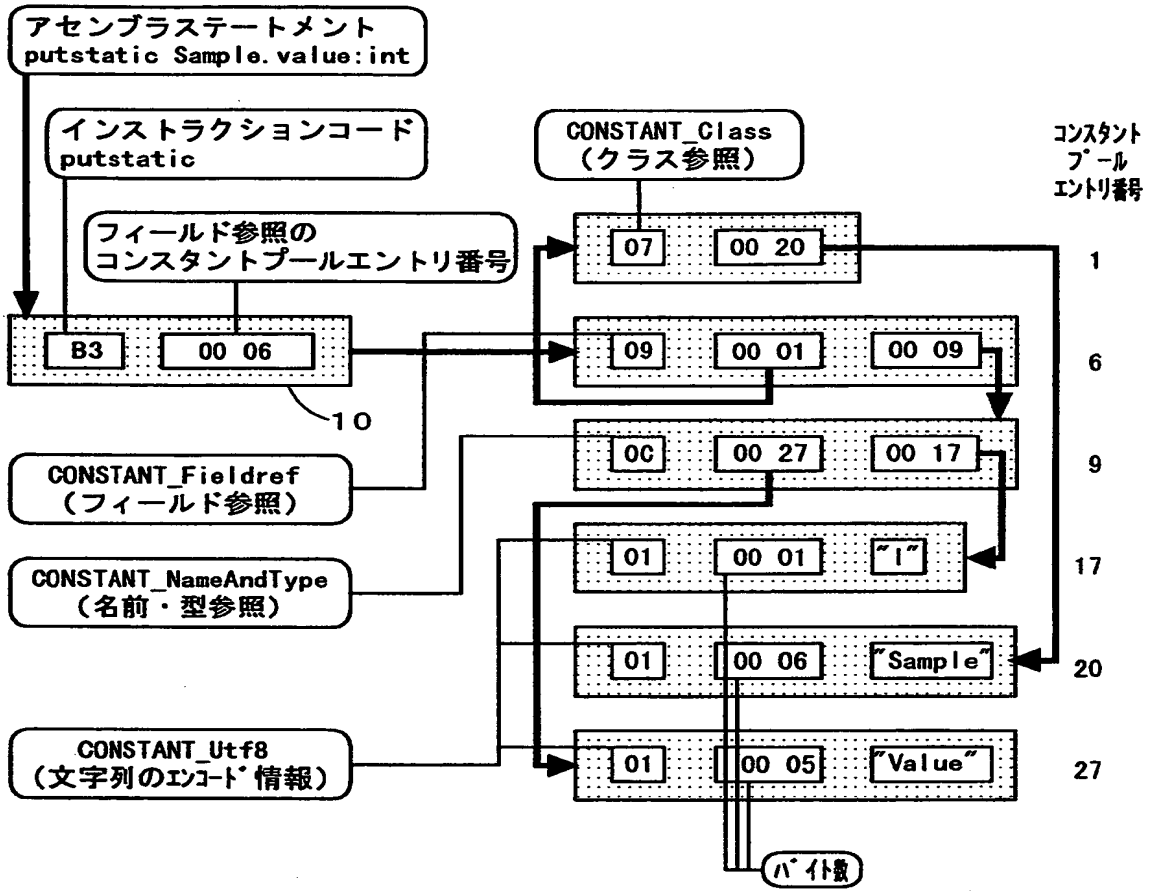

スタート
ST4－1
未解決の「フィールド参照」用の情報を抽出
ST4－2
クラス名を取得
ST4－3
クラステーブルに登録済のクラスか？
NO
YES
ST4－4
クラステーブル，フィールドテーブルへの登録
ST4－5
クラステーブルインデックスを取得
ST4－6
変数名・型名を取得
ST4－7
クラス名，変数名，型名の組合わせが一致するフィールド情報はあるか？
NO
YES
ST4－11
リンクエラー
ST4－8
一致するフィールド情報のフィールドテーブルインデックスを取得
ST4－9
ﾌｨｰﾙﾄﾞﾃｰﾌﾞﾙｲﾝﾃﾞｯｸｽ，ｸﾗｽﾃｰﾌﾞﾙｲﾝﾃﾞｯｸｽをｸﾗｽﾌｧｲﾙ内に書込み
ST4－10
未解決の「ﾌｨｰﾙﾄﾞ参照」用の情報ありか？
YES
NO
エンド

【図５】

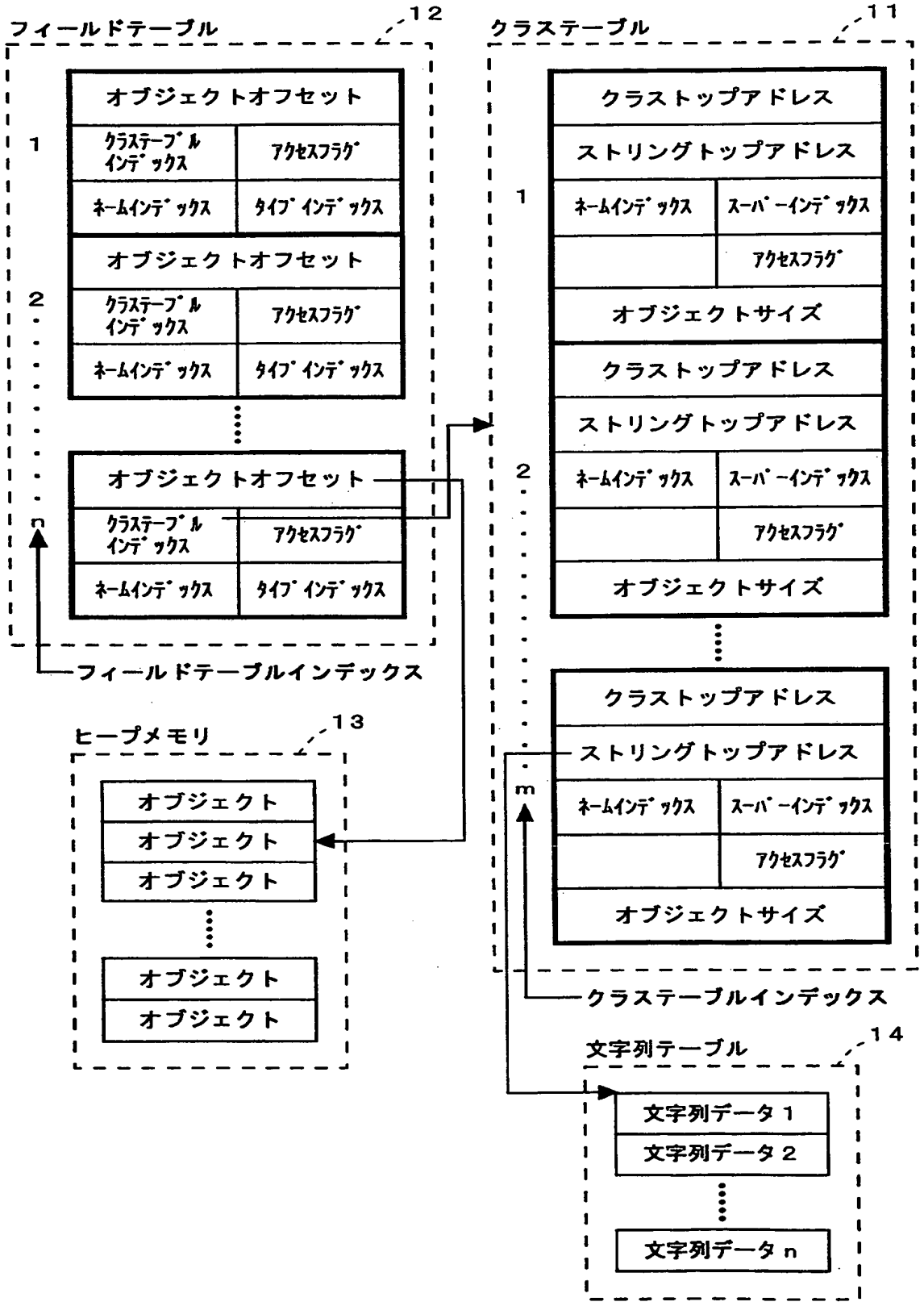


出証特２００１－３０２５８７６

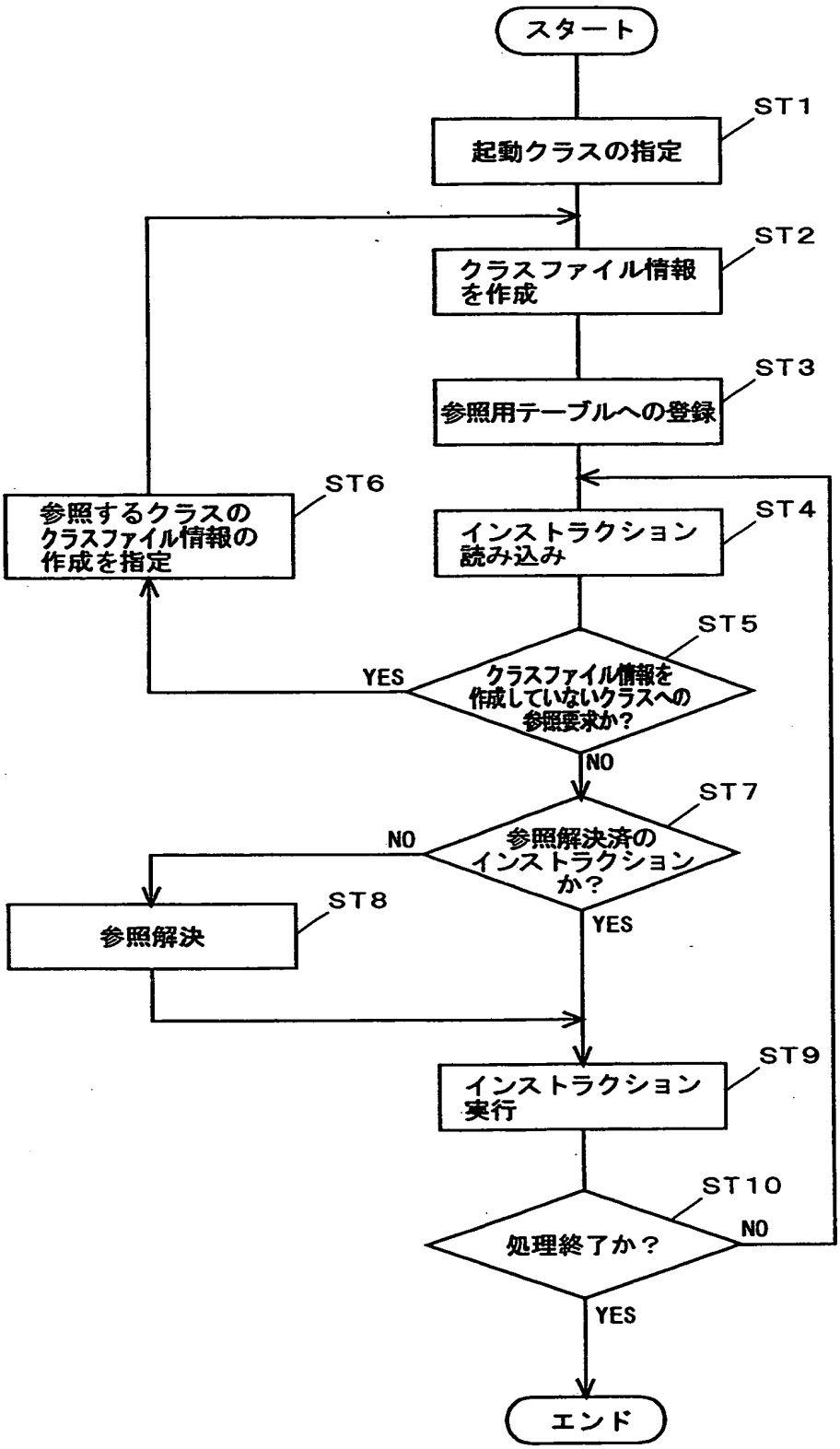
【図６】


アセンブラステートメント
putstatic Sample.value:int
インストラクションコード
putstatic
フィールド参照の
コンスタントプールエントリ番号
B3
00 06
10
CONSTANT_Fieldref
（フィールド参照）
CONSTANT_NameAndType
（名前・型参照）
CONSTANT_Utf8
（文字列のエンコード情報）
CONSTANT_Class
（クラス参照）
コンスタント
プール
エントリ番号
07
00 20
1
09
00 01
00 09
6
0C
00 27
00 17
9
01
00 01
"I"
17
01
00 06
"Sample"
20
01
00 05
"Value"
27
バイト数

【図７】


フィールドテーブル
12
オブジェクトオフセット
クラステーブルインデックス
アクセスフラグ
ネームインデックス
タイプインデックス
1
2
n
フィールドテーブルインデックス
クラステーブル
11
クラストップアドレス
ストリングトップアドレス
スーパーインデックス
オブジェクトサイズ
m
クラステーブルインデックス
ヒープメモリ
13
オブジェクト
文字列テーブル
14
文字列データ１
文字列データ２
文字列データｎ

【図８】


スタート
ST1
起動クラスの指定
ST2
クラスファイル情報を作成
ST3
参照用テーブルへの登録
ST4
インストラクション読み込み
ST5
クラスファイル情報を作成していないクラスへの参照要求か?
YES
NO
ST6
参照するクラスのクラスファイル情報の作成を指定
ST7
参照解決済のインストラクションか?
NO
YES
ST8
参照解決
ST9
インストラクション実行
ST10
処理終了か?
NO
YES
エンド

【書類名】　　要約書

【要約】

【課題】　プログラムの実行時間の安定化ならびに高速化を実現する。

【解決手段】　プログラムの実行に先立ち、フィールド参照を要求するインストラクション１０にリンクする参照用情報を用いてそのフィールド参照を解決しておき、その参照解決結果を示すフィールドテーブルインデックスを、インストラクション１０のオペランドに書き込まれたエントリ番号６のコード情報内に保存する。インストラクション１０の実行時は、前記オペランドのエントリ番号から解決されたフィールドテーブルインデックスを取得することにより、メモリ内のアクセス先のアドレスが特定される。

【選択図】　図２

認定・付加情報

特許出願の番号	特願２０００－０６９１１３
受付番号	５００００２９７０６６
書類名	特許願
担当官	第七担当上席　００９６
作成日	平成１２年　３月１４日

＜認定情報・付加情報＞

【提出日】　平成12年 3月13日

次頁無

出　願　人　履　歴　情　報

識別番号　　〔０００００２９４５〕

１．変更年月日　　１９９０年　８月２８日
［変更理由］　新規登録
住　所　京都府京都市右京区花園土堂町１０番地
氏　名　オムロン株式会社

２．変更年月日　　２０００年　８月１１日
［変更理由］　住所変更
住　所　京都市下京区塩小路通堀川東入南不動堂町８０１番地
氏　名　オムロン株式会社